Acknowledgements

Some of the slides used in this course originate with (or are modifications of) slides from colleagues at other universities. I’d like to gratefully acknowledge their contributions and thank them for generously sharing their presentation materials. These include:

- Malte Helmert
- Gabriele Roeger
- Thomas Keller
- Bernard Nebel
- Jussi Rintanen
- Patrick Haslum

Dana Nau’s lecture slides for the textbook *Automated Planning*, licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License: [http://creativecommons.org/licenses/by-nc-sa/2.0/](http://creativecommons.org/licenses/by-nc-sa/2.0/)

Shakey the Robot (1972)

https://www.youtube.com/watch?v=7bsEN8mwUB8


* RL = Reinforcement Learning
What is planning?

Automated Planning (Pithy Definition)
“Planning is the art and practice of thinking before acting.”
— Patrik Haslum

Automated Planning (More Technical Definition)
“Selecting a goal-leading course of action based on a high-level description of the world.”
— Jörg Hoffmann

Domain-Independence of Automated Planning
Create one planning algorithm that performs sufficiently well on many application domains (including future ones).

General problem solving

Wikipedia: General Problem Solver
General Problem Solver (GPS) was a computer program created in 1959 by Herbert Simon, J.C. Shaw, and Allen Newell intended to work as a universal problem solver machine.

Any formalized symbolic problem can be solved, in principle, by GPS. […]

GPS was the first computer program which separated its knowledge of problems (rules represented as input data) from its strategy of how to solve problems (a generic solver engine).
Autonomous Agents for Space Exploration

- Autonomous planning, scheduling, control
  - NASA: JPL and Ames
  - Remote Agent Experiment (RAX)
  - Deep Space 1
  - Mars Exploration Rover (MER)

Robots and Other Autonomous Systems

There is a ROS (Robot OS) interface for several of the planners we’ll study!

Not necessarily embodied!

Manufacturing Automation

Sheet-metal bending machines - Amada Corporation

Disaster management
Other Applications

- **Scheduling with Action Choices & Resource Requirements**
  - Problems in supply chain management
  - HSTS (Hubble Space Telescope scheduler)
  - Workflow management
- **Air Traffic Control**
  - Route aircraft between runways and terminals. Crafts must be kept safely separated. Safe distance depends on craft and mode of transport. Minimize taxi and wait time.
- **Character Animation**
  - Generate step-by-step character behaviour from high-level spec
- **Plan-based Interfaces**
  - Dialogue management (plan a dialogue to convey something)
  - Plan recognition

Other Applications (cont.)

- **Web Service Composition**
  - Compose web services, and monitor their execution
  - Many of the web standards have a lot of connections to action representation languages
    - BPEL, BPEL-4WS allow workflow specifications
    - DAML-S allows process specifications
- **Business Process Composition /Workflow Management**
  - Including Grid Services/Scientific Workflow Management
- **Genome Rearrangement**
  - The relationship between different organisms can be measured by the number of “evolution events” (rearrangements) that separate their genomes
  - Find shortest (or most likely) sequence of rearrangements between a pair of genomes

Other Applications (cont.)

- **Narrative generation**
- **Narrative understanding**
- **Software/Program synthesis**
- **Automated diagnosis**
- **Intelligent tutoring systems**
  - …
Model-based vs. Data-driven planning

Model-based Approaches
- Know the inner workings of the world & do a form of causal reasoning (via search) to find a plan
- Require less computation
- Taskable
- Verifiable
- Transparent, Explainable
- Good in environments where you cannot “explore” and you don’t have data or a simulator

but
- You may not have the models
  - Where do they come from?
  - Are they accurate?
- Models are “models” – approximations of the real world
If you have reasonable models, consider using/leveraging them in some way

Model-based vs. Data-driven planning

Data-driven Approaches
- Rely on data to construct a plan/policy. E.g.,
  - Model-based Reinforcement Learning (RL)
  - Model-free RL
- Perhaps learning a model or a reward function along the way
- No reliance on inaccurate models – just use data
  - Not generally taskable – maximizing expected cumulative reward
  - Can be difficult to transfer
  - Not generally transparent, explainable
  - Can require a lot of data
  - Harder to verify for safety or other properties
- Must have a simulator (a model!) or experiment in the environment (which could be unsafe or time consuming)

Learning objective for this course:
- How do we leverage the power of data in model-based sequential decision making?
- How do we leverage the power of models in data-driven sequential decision making?

(We’ll see some examples in the course.)

Example: Giving Advice to an RL Agent*

Advice: for every key in the map, get it and then go to a door; avoid nails and holes; get all the cookies.

Components of a planning problem

Agent: single agent or multi-agent

State: complete or incomplete (logical/probabilistic)

state of the world and/or agent's state of knowledge

Actions: world-altering and/or knowledge-altering (e.g. sensing)
deterministic or non-deterministic (logical/stochastic)

Goal Condition: satisficing or optimizing
final-state or temporally extended/control knowledge/script
optimizing: preferences or cost or utility or ...

Reasoning: offline or online (fully observable, partially observable)

Plans: sequential, partial order, conformant, contingent, conditional
(controller or policy)

Different classes of planning problems

Varying components of the planning problem specification yields different classes of problems. E.g.,

dynamics: deterministic, nondeterministic, stochastic
observability: full, partial, none
horizon: finite, infinite
objective requirement: satisfying, optimizing

...
Different dynamics

**Deterministic dynamics**
Action + current state uniquely determine successor state.

**Nondeterministic dynamics**
For each action and current state there may be several possible successor states.

**Probabilistic dynamics**
For each action and current state there is a probability distribution over possible successor states.

Analogy: deterministic versus nondeterministic automata

Deterministic dynamics example
Moving objects with a robotic hand: move the green block onto the blue block.

Nondeterministic dynamics example
Moving objects with an unreliable robotic hand: move the green block onto the blue block.

Probabilistic dynamics example
Moving objects with an unreliable robotic hand: move the green block onto the blue block.

Stochastic/probabilistic dynamics example
Moving objects with an unreliable robotic hand: move the green block onto the blue block.
### Different observability

<table>
<thead>
<tr>
<th>Full observability</th>
<th>Partial observability</th>
<th>No observability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations determine current world state <strong>uniquely</strong>.</td>
<td>Observations determine current world state <strong>only partially</strong>: we only know that current state is one of several possible ones.</td>
<td>There are <strong>no observations</strong> to narrow down possible current states. However, can use knowledge of <strong>action dynamics</strong> to deduce which states we might be in.</td>
</tr>
</tbody>
</table>

**Consequence:** If observability is not full, must represent the **knowledge** an agent has.

### What difference does observability make?

- **Camera A**
- **Camera B**

### Different objectives

1. Reach a goal state.
   - **Example:** Earn 500 Euros.
2. Stay in goal states indefinitely (infinite horizon).
   - **Example:** Never allow bank account balance to be negative.
3. Maximize the probability of reaching a goal state.
   - **Example:** To be able to finance buying a house by 2028 study hard and save money.
4. Collect the maximal **expected** rewards/minimal expected costs (infinite horizon).
   - **Example:** Maximize your future income.
5. ...

### Different objectives (cont.)

- **Final state properties (goals)**
- Optimization of final state properties (goals) wrt
  - Preferences
  - Probabilities
  - Cost
  - Makespan/plan length
- Temporally extended
  - goals
  - Preferences
- **Sketches**
  - Hierarchical Task Networks
  - Golog
  - LTL
  - automata
- **Rewards**
  - Markovian
  - non-Markovian
Different solutions

Form
- Sequential plans
- Conditional plans
- Policy

Properties
- Finite
- Infinite

Criteria
- Satisficing
- Optimal
  - Cost
  - Makespan
  - Preferences (Objective function)
  - Expected Cumulative Reward
  - ...

These differing properties induce different classes of planning problems with differing characteristics

Different Classes Planning Problems

dynamics: deterministic, nondeterministic, stochastic
observability: full, partial, none
horizon: finite, infinite
objective requirement: satisfying, optimizing
...
- classical planning
- conditional planning with full observability (FOND)
- conditional planning with partial observability (POND)
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, nondeterministic, stochastic
observability: full, partial, none
horizon: finite, infinite
objective requirement: satisfying, optimizing
...
- classical planning
- conditional planning with full observability (FOND)
- conditional planning with partial observability (POND)
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning
Different Classes Planning Problems

**dynamics:** deterministic, **nondeterministic**, stochastic

**observability:** full, partial, none

**horizon:** finite, infinite

**objective requirement:** satisfying, optimizing

...  
  - classical planning  
  - conditional planning with full observability  
  - **conditional planning with partial observability**  
  - conformant planning  
  - markov decision processes (MDP)  
  - partial observable MDP (POMDP)  
  - preference-based/over-subscription planning

---

**dynamics:** deterministic, **nondeterministic**, stochastic

**observability:** full, partial, none

**horizon:** finite, infinite

**objective requirement:** satisfying, optimizing

...  
  - classical planning  
  - conditional planning with full observability  
  - conditional planning with partial observability  
  - **conformant planning**  
  - markov decision processes (MDP)  
  - partial observable MDP (POMDP)  
  - preference-based/over-subscription planning

---

**dynamics:** deterministic, **nondeterministic**, stochastic

**observability:** full, partial, none

**horizon:** finite, infinite

**objective requirement:** satisfying, optimizing

...  
  - classical planning  
  - conditional planning with full observability  
  - conditional planning with partial observability  
  - conformant planning  
  - markov decision processes (MDP)  
  - partial observable MDP (POMDP)  
  - preference-based/over-subscription planning
### Different Classes Planning Problems

**dynamics:** deterministic, nondeterministic, stochastic  
**observability:** full, partial, none  
**horizon:** finite, infinite  
**objective requirement:** satisfying, optimizing  

- classical planning  
- conditional planning with full observability  
- conditional planning with partial observability  
- conformant planning  
- markov decision processes (MDP)  
- partial observable MDP (POMDP)  
- preference-based/over-subscription planning

### Other Dimensions

**dynamics:** deterministic, nondeterministic, stochastic  
- explicit time, implicit time  
- instantaneous, durative  
- continuous, discrete, hybrid  
**agents:** multi-agent  
**perception:** perfect, noisy  
**horizon:** finite, infinite  
**objective requirement:** satisfying, optimizing  
**objective form:** final-state goal, temporally-extended goal, control knowledge, hierarchical task network (HTN), script/program (Golog)  
**plan form:** sequential plan, partial order plan, controller, policy, generalized plan, program...  

- classical planning  
- conditional planning with full observability  
- conditional planning with partial observability  
- conformant planning  
- markov decision processes (MDP)  
- partial observable MDP (POMDP)  
- preference-based/over-subscription planning

### Three Main Classes of Planners

1. Domain-specific  
2. Domain-independent  
3. Domain-customizable

* Ghallab, Nau, and Traverso’s use “configurable” (which I don’t like)  
Also called “Domain-specific” or “Knowledge-Based”

### 1. Domain-Specific Planners

- Made or tuned for specific domain  
- Won’t work well (if at all) in any other domain  
- Many successful real-world planning systems work this way
2. Domain-Independent Planners

- In principle, a domain-independent planner works in any planning domain
- Uses no domain-specific knowledge except the definitions of the basic actions

Restrictive Assumptions

- **A0: Finite system:**
  - finitely many states, actions, events
- **A1: Fully observable:**
  - the controller always knows the system’s current state
- **A2: Deterministic:**
  - each action has only one outcome
- **A3: Static** (no exogenous events):
  - changes only occur as the result of the controller’s actions
- **A4: Attainment goals:**
  - a set of goal states \( S_g \)
- **A5: Sequential plans:**
  - a plan is a linearly ordered sequence of actions \( (a_1, a_2, \ldots, a_n) \)
- **A6: Implicit time:**
  - Actions are instantaneous (have no duration)
- **A7: Off-line planning:**
  - planner doesn’t know the execution status

Classical Planning

- Classical planning: all 8 restrictive assumptions
  - Offline generation of action sequences for a deterministic, static, finite system, with complete knowledge, attainment goals, and implicit time
  - Reduces to the following problem:
    - Given \((\Sigma, s_0, G)\)
    - Find a sequence of actions \((a_1, a_2, \ldots, a_n)\) that produces a sequence of state transitions \((s_1, s_2, \ldots, s_n)\) such that \(G\) is in \(s_n\)
  - This is just path-searching in a graph
    - Nodes = states
    - Edges = actions
- *Is this trivial?*
Classical Planning as state-based search

Why is classical planning difficult?

Classical planning is computationally challenging:
- number of states grows exponentially with description size when using (propositional) logic-based representations
- provably hard (PSPACE-complete)
  \( \sim \) we prove this later in the course

Problem sizes:
- Seven Bridges of Königsberg: 64 reachable states
- Rubik’s Cube: \( 4.325 \cdot 10^{19} \) reachable states
  \( \sim \) consider 2 billion/second \( \sim \) 1 billion years
- standard benchmarks: some with \( > 10^{200} \) reachable states

Why is planning difficult?

- solutions to classical planning problems are paths from an initial state to a goal state in the transition graph
  - efficiently solvable by Dijkstra’s algorithm in \( O(|V| \log |V| + |E|) \) time
  - Why don’t we solve all planning problems this way?
- state spaces may be huge: \( 10^{10}, 10^{100}, 10^{1000}, \ldots \) states
  - constructing the transition graph is infeasible!
  - planning algorithms try to avoid constructing whole graph
- planning algorithms are often much more efficient than obvious solution methods constructing the transition graph and using e.g. Dijkstra’s algorithm

…so in this course …
What will you learn in this course?

- **Theoretical background** for planning
  - Formal problem definition
  - Basic theoretical notions (e.g., normal forms, progression, regression)
  - Computational complexity of planning
- **Algorithms** for planning:
  - Based on heuristic search
  - Based on exhaustive search with logic-based data structures such as BDDs (if time permits)
  - Many of these techniques are applicable to problems outside AI as well.
- **Hands-on experience** with a classical planner (probably)

What will you learn in this course?

**Spectrum of Planning and Learning (Sequential Decision Making) Algorithms**

- Classical planning
- Nondeterministic planning (FOND)
- Stochastic planning
- MDPs and POMDPs
- Reinforcement learning (RL)

Incorporating into these algorithms

- Rich objectives (beyond final state goals)
- Different models of state

Understanding diverse applications for these algorithms (beyond planning)

- ...

For next day

**Admin**

- Complete the doodle poll (early next week)
- Ensure you’re on piazza
- Get in touch if you have further questions about the course

**Course**

1. Read Hoffman paper “Everything you wanted to know about planning…” (linked on web page)
2. Read Chapter 1 of Geffner and Bonet (online)
3. Familiarize yourself further with PDDL
4. Try out a planning.domain -- run a planning problem
And to help with 3 and 4 ...

A quick overview of PDDL

Fun with search and RL!

Kiva Robots - https://www.youtube.com/watch?v=3eQAFVetNGI

AlphaGo - https://youtu.be/g-dKXOlsf98?t=145

Arcade Learning Environment (ALE)
https://www.youtube.com/watch?v=V1eYniJ0Rnk - Breakout
https://www.youtube.com/watch?v=5WXVJ1A0k6Q - Seaquest
https://www.youtube.com/watch?v=P-603qPMkSg - Width-based planners

Reinforcement Learning for Soccer Goalie
https://www.youtube.com/watch?v=ClF2SBVY-J0

Robot Path Planning and Arm Manipulation
https://www.youtube.com/watch?v=Sv-J37zcLU4